DOCUMENT RESUME

ED 265 222 TM 860 089

AUTHOR McArthur, David L.
TITLE Diagnostic Testing Project.
INSTITUTION California Univ., Los Angeles. Center for the Study of Evaluation.
SPONS AGENCY National Inst. of Education (ED), Washington, DC.
PUB DATE Apr 85
GRANT G-85-0001-P1
NOTE 37p.; Some pages have small print.
PUB TYPE Reports - Descriptive (141) -- Computer Programs (101)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Adaptive Testing; *Computer Assisted Testing; *Computer Software; *Diagnostic Tests; Elementary Secondary Education; Individual Testing; Item Banks; Microcomputers; Psychometrics; *Test Construction; Testing Programs
IDENTIFIERS *DX System (Computer Software)

ABSTRACT

This paper presents an explanation of the program design of the three separate Pascal-language programs which comprise UCLA's Diagnostic Testing Package "DX." The three parts of the DX test package are three closely interrelated programs--Editest, Runtest, and Summary. Editest serves the following functions: (1) the creation of a suitable direct-access file of textual material for each test module; (2) the automatic creation of special files known as cue-files, which provide the requisite program pointers that allow the testing process to be adaptive, and error types to be found within a test; (3) the editing of the direct-access file of text using a simple line-oriented editor; and (4) the review of a specified direct-access file of test materials one screen at a time. The program Runtest is the computive central manager of the adaptive testing sequence, and the student response file is evaluated by the Summary program. Flow charts for each program and test administration are given. Samples are included showing: (1) test item layout; (2) location cue files; and (3) test performance summaries. The Pascal listings for each of the three programs are also appended. (LMO)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

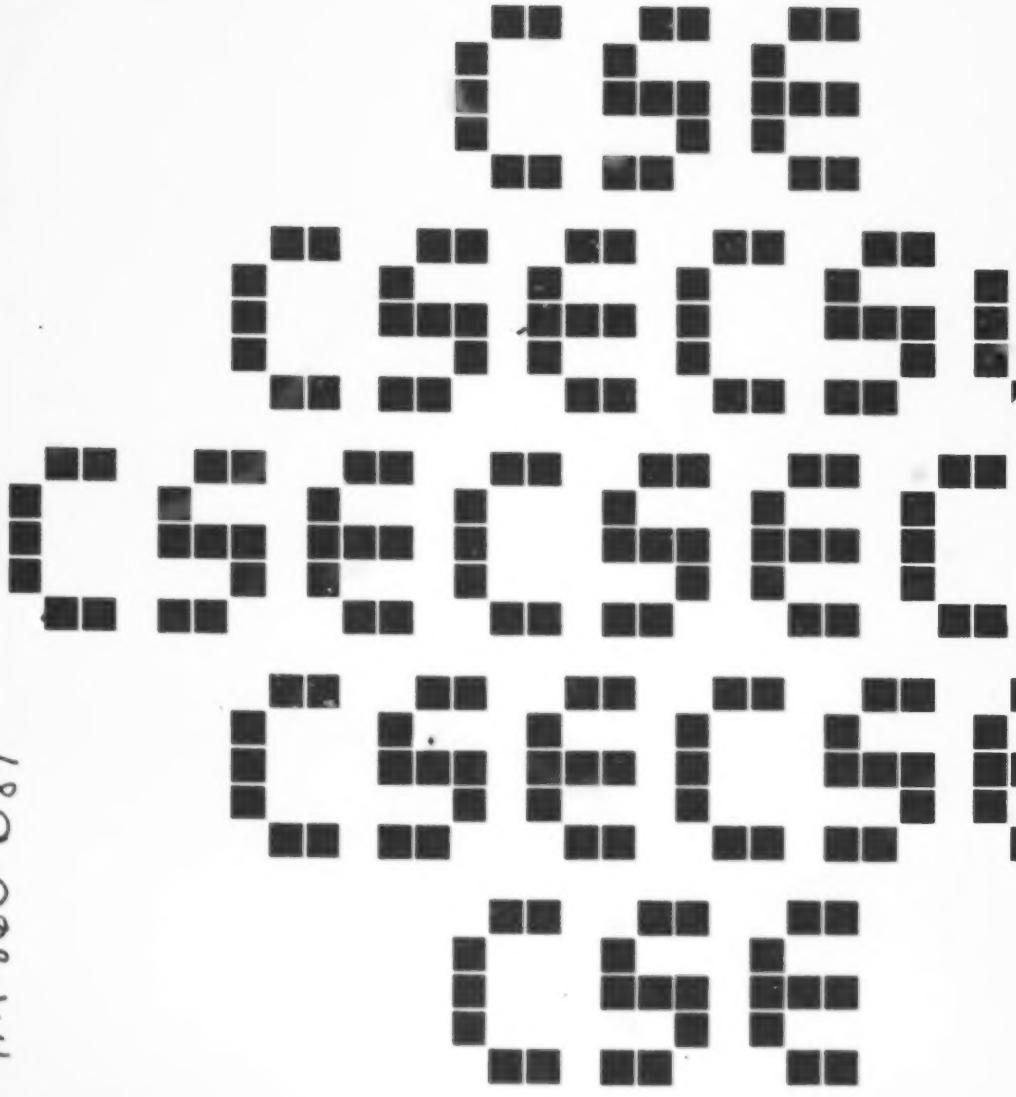
Center for the Study of Evaluation

UCLA Graduate School of Education
Los Angeles, California 90024

DELIVERABLE - APRIL 1985

DIAGNOSTIC TESTING PROJECT

ED265222



U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it.

☑ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

J.C. Beer

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

TM 860 089

DELIVERABLE - APRIL 1985

DIAGNOSTIC TESTING PROJECT

David L. McArthur
Project Director

Grant Number
NIE-G-85-0001
P1

Center for the Study of Evaluation
Graduate School of Education

BEST COPY

The project presented or reported herein was supported pursuant to a grant from the National Institute of Education, Department of Education. However, the opinions expressed herein do not necessarily reflect the position or policy of the National Institute of Education and no official endorsement by the National Institute of Education should be inferred.

BEST COPY

INDEX

"Microcomputer software for Diagnostic Testing"

Appendix A Program flow for EDITTEST

Appendix B Program flow for RUNTEST

Appendix C Program flow for SUMMARY

Appendix D Sample schematic flow for test administration

Appendix E Program EDITTEST - Pascal listing

Appendix F Program RUNTEST - Pascal listing

Appendix G Program SUMMARY - Pascal listing

Appendix H Sample text showing test item layout

Appendix I Sample location cue files>

Appendix J Sample test performance summaries

Microcomputer Software for Diagnostic Testing

David L. McArthur

Center for the Study of Evaluation

April 30, 1985

The following is an explanation of the logical underpinnings of the three separate Pascal-language programs which comprise UCLA's Diagnostic Testing Package "DX." The intent of the program designers has been to provide a system for computer managed adaptive diagnostic testing which is efficient, flexible, and modifiable in terms of many of its key operations. Within the DX framework, an inexperienced computer user could make new microcomputer-managed tests which use features similar to the tests used by the UCLA team in the development of DX, while an moderately experienced computer user could create computer-directed testing scenarios which are quite distinct from those prototypes.

The three parts of the DX test package are three closely interrelated programs -- Edittest, Runtest, and Summary. The first program, Edittest, serves the following functions:

a) the creation of a suitable direct-access file of textual material for each test module;

b) the automatic creation of special files known as "cue-files," which provide the requisite program pointers that

BEST COPY

allow the testing process to be adaptive, and error types to be found within a test;

c) the editing of the direct-access file of text using a simple line-oriented editor;

d) the review of a specified direct-access file of test materials one screen at a time (one test item plus one or more test questions).

The program Runtest is the computer's central manager of the adaptive testing sequence. It administers one or more tests, halting to accept the student's responses, and delivering feedback (if delivery of feedback has been requested) as to whether the student was right or wrong, recognizing acceptable response entries and rejecting others. At the same time it is tracking the student's performance against internal constants, deciding whether to continue the testing, to move up or down in item difficulty, or to exit to the next test module. It is also preserving a file of the student's responses, one for each separate test taken.

The student response file is evaluated by the program Summary, which analyzes the responses both by conventional statistics (% correct overall and % correct within different ranges of item difficulty levels) and by error patterns, which it compares to the error cue-file for that test. It presents (and prints if requested) an individual performance summary annotated with the type(s) of errors made by the student and the number of the first test item for which that error was applied.

BEST COPY

Efficiency of the programming is generated in three ways. The first of these is the use of the programming language Pascal, which from its beginnings has been regarded as a high-quality language suitable for rigorous logical sequencing, readily transportable between different machines using p-System (a well-known microcomputer operating system) and, because of its conceptual structure, relatively easy to review and understand. The second feature which contributes to efficiency is the use of direct-access file structures, which allow the program to select different pieces of text from a test file with minimum delay. That is, the operation of the testing session, even during its most complex phases, seldom has to wait more than a second or two for a completely fresh display of text. The third facet of the programming which enhances efficiency is the use of "windowing," a system of bringing material to particular locations on the video screen, so that, for example, one test passage may be followed by several test questions but only the text of the questions need be updated.

Flexibility of the DX package is addressed in four distinct ways. First, the programs have been designed to fit a wide range of testing protocols, excluding in general only those tests which require free response. Second, the placing of test materials into the appropriate files is restricted only by the number of text lines which may fit on a single video screen, and by the character generation set (generally conventional

BEST COPY

alphanumeric symbols) available in a particular computer. The input and editing of these materials may occur either under the computer's usual p-system text-editing facility or with a line-at-a-time editor supplied in Edittest. The materials also may be reviewed serially. The third aspect of flexibility is the program's capacity to accept error coding of up to nine different types and to recognize those types of errors separately by test. That is, multiple tests can be administered in a single session; the results of each test can be keyed to separate error indicators. The fourth contribution to flexibility is that the primary logic commands which drive the testing operation (See Appendix A) are contained in a handful of statements located together, and a special subset of program constants which govern how the test proceeds are also located together. Both of the these subsets may be easily accessed in the sourcecode.

The DX Package can be modified in any of three ways. The easiest way is to type in new test materials which obey the same principles (described below) as the two test prototypes used in the development of the DX package at UCLA. Parenthetically, even at this stage, the new test may utilize different error coding and different numbers of multiple choice responses, as well as entirely fresh text materials. The second way to modify the package is to reset one or more of the governing constants mentioned earlier (and detailed below) and/or to extend or reduce the program's search for additional test modules to administer

BEST COPY

after the student completes the first test module. The third route to modification of the program could occur in the subset of logical statements which manage the testing; this task requires knowledge of Pascal and of the program's present flow structure. However, because of Pascal's inherent rigor, such changes are relatively simple conceptually and not difficult to program. In each instance where a modification could be introduced, sourcecode has been annotated accordingly (See Appendix B).

The core set of principles which shape the program's management of an individual testing session can be summarized as follows:

a) After accessing appropriate files and initializing variables, the program begins the testing sequence using an item stem and question drawn at a preselected level of item difficulty;

b) The program administers test items one at a time, and evaluates the student's responses. When the tallies of the student's correct and incorrect responses at the present level of test item difficulty exceed a preset critical value, the program moves up or down one level in item difficulty. If no change in level is warranted, the next item within the same level is administered;

c) If a change in item difficulty level is warranted, but no additional levels exist, the program moves to the next test. If additional levels exist, the program moves to the first item within that level.

BEST COPY

In each instance, decisions are made with the assumption that a predominantly nonlinear testing sequence will best allow the test to be fitted to the evident capabilities of the student. Because of its adaptive scheme, seldom will a single student sit through every test item.

Governing constants within the DX package are values which the program logic uses to check the student's progress through the testing session. These include the level of item difficulty from which to select the first test item, the number of correct responses required within a given item difficulty level in order to skip any remaining items still present at that level and move to the next lower or higher level for the next test item, and the number of questions to be found within a given item level. Note that distinctions made using the phrase "item difficulty" can be made using any suitable psychometric criterion. As mentioned earlier, each of these parameters can be changed without conceptual modifications to the DX package.

A portion of one prototype test and its associated cue files have been annotated to show how the program Runtest operates with a given test item and questions (See Appendix C).

The DX system was developed on a SAGE IV 68000-based microcomputer, for applications using the Apple II-plus and IIe 6802-based machines. System requirements include the addition of one memory card to bring the machine to 64K of RAM, two disk drives, and video monitor. Printer is optional. The Pascal

BEST COPY

operating system, DX programs and error cue files are accessed on the first disk, while the direct-access test files and associated cue-files are placed on the second drive, to which student data files are also written during the testing session. File sizes are as follows: random access files occupy about five blocks per test question, while cue-files are four blocks long and student response files about six to eight blocks apiece.

Experience demonstrates that students familiar with reading text materials on a video screen progress somewhat faster and with rather more confidence then students whose computer experience is limited. Additionally, the latter group may require an increase in the program control parameter which determines the length of built-in pauses (for example, at the conclusion of the opening instructions or the delivery of right-wrong feedback).

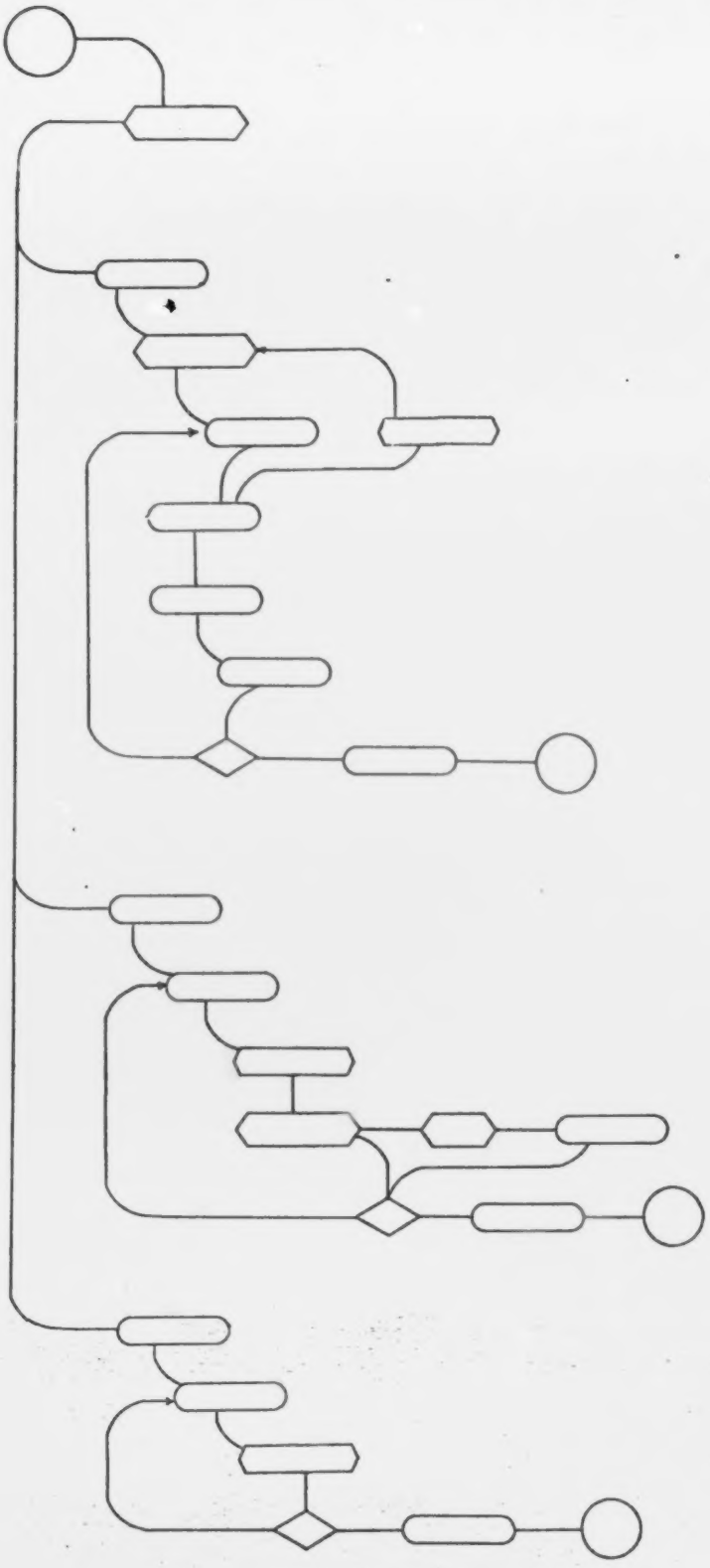
Attempts have been made to ensure that unexpected or illegal responses made by a student are not inadvertantly counted as correct or incorrect; a student deliberately or accidently hitting "f" or "7" --when the allowable choices are only "a" through "e"-- is allowed three such keystrokes before the computer scores that test question as "No response" and moves on. The system construtcs its file of student results under a directory entry made up of first initial, last name, and first letter of the test. A student with the same first initial and last name as one already represented on the second disk is automatically assigned the number "2" (or "3" or "4" as necessary) for the response

BEST COPY

file's directory entry. These response files are compatible with UCLA's Statpak microcomputer statistical analysis package.

Minor refinements in the present package of programs are anticipated following completion of the present course of field trials being conducted with fourth, fifth and sixth grade pupils in the Los Angeles Unified School District.

Program flow for EDITTEST



Start

Select from menu

Create test:

Make file allocations

Is text for test available?

Get text from file OR Enter text from keyboard

Determine location cues

Save text entry to file

Save location cue to file

More? Close files Stop

Modify test:

Access files for test and location cues

Determine location

Show 1 line of text

Need changes? Type new line Replace old lin

More? Close files Stop

Review test:

Access files for test and location cues

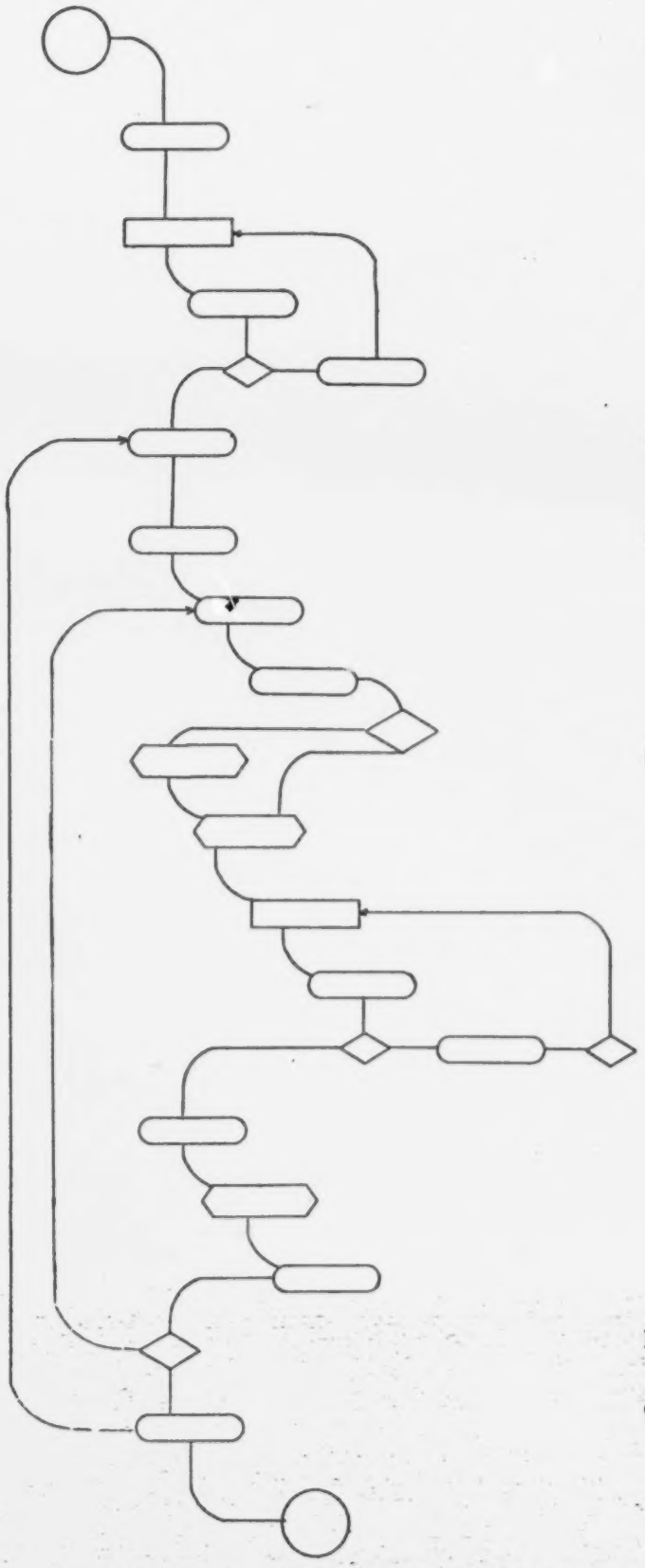
Determine location

Show 1 screen of text

More? Close files Stop

BEST COPY

Program flow for RUNTEST



Start

Initialize internal variables

Ask for student's name

Make file allocation for this student

OK? Need to ask again

Access files for test and location cues

Show general instructions to student

Determine location of item text

Determine location of question text

Same item as already shown?

Show item

Show question and response options

Get student's response

Check response against options

OK? Need to respond again 3rd try ?

Process student's response

Show feedback to student

Pause for n seconds

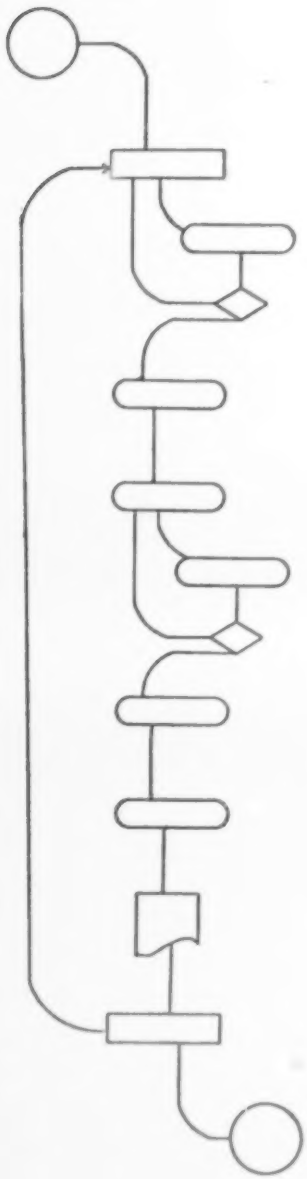
More questions and items?

Close files and move to next test packet

Stop

BEST COPY

Program flow for SUMMARY



Start

Which student's performance to summarize?

Check directory of performance files

OK?

Access necessary files

Read data

Compute response information

More?

Access error set file

Compute error indicators

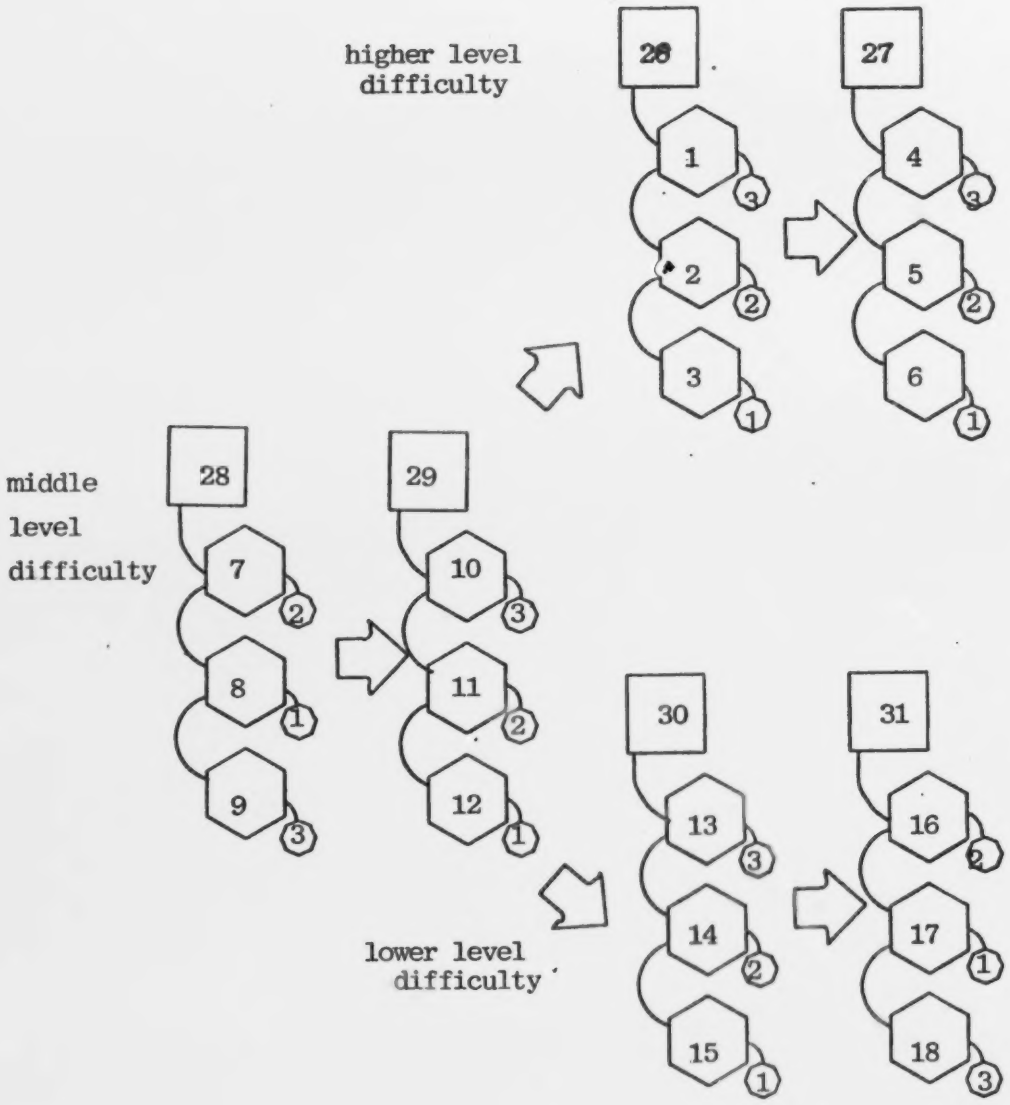
Print summary to screen and/or hardcopy

Continue with next student?

Stop

Sample schematic flow of test administration

Reading Comprehension Test



BEST COPY

Program EDITTEST

© 1985 Regents of the University of California

```
  program EDITTEST;                                  (* edition 12-apr-85 *)
(* microcomputer-managed diagnostic testing  UCLA:  NIE
       Grant No. G-84-0112,  D.L.McArthur and C.R.Morris.......*)

  CONST
        Maxlines  = 24;                    (*max # lines per item*)
        Maxitem   = 18;                          (*max # of items*)
        toplevel  = 3;                     (*highest level in test *)
        itemsperlev= 6;                     (** items per level *)
        Disc      = '#5:';                    (*data disk is #5  *)
        Screensize= 23;                      (*screen has 23 lines max*)

        dash = '-----------------';
  TYPE item = record
    material  :array[1..Maxlines] of string[80];end;  (*item goes into array*)
  CS  = set of char;
  VAR   iocheck,jtxt,lineknt,count, i, j, k,questot   :integer;
        NumofItem, recno, itemn, questn, leveln, quesloc  :integer;
        keycode:    char;  Uend   :string[3];
        code,check,place,h1,h2,h3,h4,h5     :string[1];
        temporary,filename, phrase,trk,txtsource    :string[80];
        txtonfile,getout,goodbye :Boolean;
        Questions    :file of item;
        OUT, SOURCETXT    :text;               (*end of setup*)

procedure INITIALIZE; (*initialize parameters*)
begin; with Questions^ do count:= 0;questot:=0;i:=0;
      recno:=0;itemn:=0 ;questn:=0;leveln:=toplevel;quesloc:=Maxitem+1   ;
         goodbye:=false;getout:=false;Uend:='!!!'; NumofItem:=1;
   if txtonfile then jtxt:=1;
         (*$I-*)  reset (OUT,trk);            (*if reading from existing file*)
                  iocheck:= ioresult;
         (*$I+*) end;                        (*end of initialization*)

procedure CLRSCRN; (*clear the screen completely*)
 begin;   GOTOXY(0,0);
   for k:=1 to 25 do begin; write('                                                ');
        writeln('                                                ');  end;end;


procedure INSERTTEXT (i :integer);
   begin;                                    (*bring one text line into file*)
j:=0;
if (txtonfile) then jtxt:=1;
    with Questions^ do begin;
    if txtonfile then begin;
        readln(SOURCETXT,material[i]);
        writeln (material[i]); end
   else begin;
        writeln('  Insert line # ',i,' of the text here,   or  <*>');
        readln(material[i]);
         end;
    writeln (dash,dash,quesloc);lineknt:=lineknt+1;
       if (lineknt>Screensize) then writeln(' TOO MUCH TEXT');
    phrase:=(material[i]);
    code:=copy(phrase,1,1);
    end;
  end;                                       (*end of inserting text*)
```

BEST COPY

```
procedure WRAPFILES;                                    (*wrap up open files now*)
     begin;      writeln ('   end of file');
     quesloc:=quesloc+1;i:=1;
     seek (Questions,quesloc);put(Questions);
     recno:=0;itemn:=0;questn:=0;leveln:=0;quesloc:=0;
     writeln (OUT,recno:5,itemn:5,questn:5,leveln:5,quesloc:5,' 9 9 9 9 9 9 9');

end;                                                    (*end of wrapping files*)

procedure CLOSEFILES;     (*close all open files now*)
    begin; close (Questions, lock);
      close (OUT,crunch);
      if txtonfile then close(SOURCETXT, lock);
      goodbye:=false;
    end;                                                (*end of closing files*)

procedure CHECKINPUT (var checknum  :integer);
      var input: char; output  :integer;        (*is input any good?*)
      procedure INCHECK;
      begin;
          if input = '&' then exit(program);
          output:= ORD(input) - ORD('0');
          if (output >=0) and (output <=10) then begin
              output:= ord(input) - ord('0');
              checknum:=output;end
          else begin
              writeln ('...Sorry, that character isn''t legal here...');
              writeln ('     Type in the correct number:  ');end;
   end;begin
          readln (input);INCHECK;
   end;                                                 (* end of checkinput *)

function LOOKEND  (code: string):integer;
   var found   :Boolean;  index,kk  :integer;      (* are we ended yet? *)
   begin;
         index:=1;  found:= false;
             if code = ')' then begin;
                getout:=true;
                end;
             if (code = '!') then begin;
                lookend:=1;
                goodbye:=true;
                end;
             if (code = '*') then begin
                  found:= true;
                  lookend:= 1;
                  kk:=length(phrase);
                    if (kk>1) then h1:=copy(phrase,2,1) else h1:='9';
                    if (kk>2) then h2:=copy(phrase,3,1) else h2:='9';
                    if (kk>3) then h3:=copy(phrase,4,1) else h3:='9';
                    if (kk>4) then h4:=copy(phrase,5,1) else h4:='9';
                    if (kk>5) then h5:=copy(phrase,6,1) else h5:='9';
                if (kk<2) then itemn:=itemn+1;
                if (kk<2) and (NumofItem>1) and (NumofItem mod itemsperlev=1)
                              then leveln:=leveln-1;
             end
             else begin
                  index:= index+1;
                  lookend:= 456
                  end;
   end;                                                 (* end of function lookend *)
```

BEST COPY

```
procedure WRITEITEM;  (* write item to proper location*)
    begin;j:=0;
        quesloc:= quesloc + 1   ;
        if txtonfile then jtxt:=1;
        if not txtonfile then begin;
           writeln(dash,dash);
           writeln('Enter * as the last line');
           writeln(dash,dash);writeln; end;
        seek (Questions, quesloc);
        with Questions^ do begin;
        repeat;
           i:=i+1;
           INSERTTEXT (i);
           until (LOOKEND(code)=1) or getout or goodbye; end;
           put (Questions);
   if goodbye then  phrase:=Uend; end;                (* end of writing item *)

procedure SETFILENAME;          (* what is file name? *)
   begin;       writeln;writeln (dash,dash,dash);
        write ('  Enter the name for this Test File :  ');
        readln (filename);
        if (filename = 'stop') or (filename = 'STOP') then begin;
        writeln ('.....ending at your request.....'); exit (program); end;
   filename:= concat(DISC,filename);      (* random-access text file *)
        trk:= concat(filename,'.Q.TEXT');      (* sequential command file *)
        end;                                    (*  end of setting filename *)

procedure DIRECTORY;                            (* brief directory of files *)
   VAR  VOLUME  :INTEGER;
     STRPNTR,I,PROG,VALUE, BYTE,BLOCK,L : INTEGER;
     ENTRY, TOTPROGS : INTEGER;
     BUFFER  :PACKED ARRAY [0..511] OF 0..255;
     PTITLE : STRING;
 BEGIN;
      ENTRY:= 0;volume:=5;
      WRITELN;
      BLOCK:= 2; BYTE:=6;
      UNITREAD (VOLUME,BUFFER,512,BLOCK,0);
      WRITELN;
      count   := BUFFER[16];
      WRITELN ('Catalog of volume #',VOLUME);
      WRITELN;
      FOR PROG:= 1 TO count    DO BEGIN
          BYTE:= BYTE + 26;
          IF BYTE > 511 THEN BEGIN
              BLOCK:= BLOCK + 1;                    (* directory cont'd.*)
              BYTE:= BYTE - 512;
              UNITREAD(VOLUME,BUFFER,512,BLOCK,0);
            END;
              ENTRY:= ENTRY + 1;
              PTITLE:= '                          ';
              L:=BUFFER[BYTE];
              IF BYTE+L>511 THEN L:=511-BYTE;
              FOR I:= 1 TO L  DO
                 PTITLE[I]:= (CHR(BUFFER[BYTE+I]));
                WRITE (PTITLE);
                IF ENTRY MOD 3 = 0 THEN WRITELN;
   END;end;                                          (* end of directory *
)
```

BEST COPY

```
procedure EXISTING;  (* what file? does it exist? *)
      begin;       writeln ('....Shall we use an existing file for input');
      writeln ('      or the keyboard to type in this new material?');
      write ('          <E>xisting file  <K>eyboard.....   ');
      readln (code);txtsource:='CONSOLE:';
           if (code='e' )or( code='E') then begin;
               txtonfile:=TRUE;
               if (txtonfile) then jtxt:=1;
               writeln ('   What is the filename for this text?');
               readln (txtsource);iocheck:=0;
                        (*$I-*)
               reset (SOURCETXT,txtsource);
                 iocheck:= ioresult;
                        (*$I+*)
                 if iocheck<>0 then begin;
                 writeln ('File not found'); DIRECTORY;exit(EXISTING);end;
             end;
      end;                                        (* end of existing check *)


procedure SEERECNUM;
   begin;                                         (* see command file *)
      read (OUT,recno,itemn,questn,leveln,quesloc,h1,h2,h3,h4,h5);
      iocheck:=ioresult; end;

procedure PUTRECNUM;
     begin;                                       (* put data into command file *)
     writeln ('---> ',recno,' ',
                NumofItem:5,itemn:5,questn:5,leveln:5,quesloc:5
                ,'  ',h1:3,h2:3,h3:3,h4:3,h5:3);
     writeln (OUT,NumofItem:5,itemn:5,questn:5,leveln:5,quesloc:5
                ,'  ',h1:3,h2:3,h3:3,h4:3,h5:3); end;

procedure ACCESSMAKE;
     var ch: char;                    (* primary routine to make test file *)
     begin; iocheck:=0; txtonfile:=FALSE;txtsource:='CONSOLE:';
        CLRSCRN;GOTOXY(0,10);
        SETFILENAME;
        INITIALIZE;getout:=true;
        (*$I-*) reset (Questions,filename);
        if ioresult>0 then iocheck:=1;
        (*$I+*)
        CLOSEFILES;
        if iocheck=0 then begin
          writeln ('   File by that name already exists.....');
             DIRECTORY;
             write   ('  Would you like to replace it?  <Y><N> ');
             readln (ch);
             if (ch<> 'y')and (ch <>'Y') then SETFILENAME;
             rewrite (OUT,trk);
             writeln (OUT,'0  This is the command sequence for ',filename);
             rewrite (Questions, filename);
         end
    else begin; rewrite (OUT,trk);                    (* accessmake cont'd *)
         writeln (OUT,'1  This is the command sequence for ',filename);
          writeln('Preparing disk...');
          for J:=1 to 800 do writeln(OUT,j);CLRSCRN;close(OUT,crunch);
           reset   (OUT,trk);
             writeln (OUT,'2  This is the command sequence for ',filename);
         rewrite (Questions, filename);
    end;
```


BEST COPY

```
   EXISTING;
       if iocheck<>0 then EXISTING;
   while (LOOKEND(code)<>0) do repeat;
       if (goodbye) then begin;WRAPFILES;CLOSEFILES;EXIT(ACCESSMAKE);end;
  recno:= NumofItem;lineknt:=0; questn:=NumofItem;
       if (getout) and (questn>questot) then begin;
             getout:=false;WRITEITEM; end;
       seek (Questions, recno);                          (*accessmake cont'd*)
       with Questions^ do begin;
       i:=0;
     if (txtonfile) then jtxt:=1;
          if (recno=0) and (not txtonfile) then begin
            writeln(dash,dash);
                        writeln('Enter * as the last line + HYPOs');
                        writeln(dash,dash);writeln end;
           repeat;
                  i:=i+1;
                  INSERTTEXT (i);
          until (LOOKEND(code)<2)or getout or goodbye     end;
       if not getout and not goodbye then begin;PUTRECNUM;
       NumofItem:=NumofItem+1;end;
      CLRSCRN;
        put (Questions);
        until (NumofItem>= Maxitem)or goodbye;WRAPFILES;
        CLOSEFILES;
end;                                    (*.....................end accessmake *)

procedure ENDYET;
              begin;  if quesloc>0 then exit(ENDYET);
              writeln(' [[[[[[[[[[[[[[[[[ end of file ]]]]]]]]]]]]]]]]]');
              goodbye:=true;CLOSEFILES;CLRSCRN;end;

procedure ACCESSCHANGE;
 var changenum: char;                   (* primary routine to modify *)
   begin; CLRSCRN; DIRECTORY; getout:=true;SETFILENAME;
    (*$I-*) reset (Questions,filename);
            iocheck:=ioresult;
            reset (OUT,trk);
    (*$I+*)
    if iocheck<>0 then begin
       writeln ('File not found'); DIRECTORY;SETFILENAME;
       reset (Questions,filename) end;
    readln(OUT,jtxt);
    repeat;
      SEERECNUM;
      if (goodbye) then begin; CLOSEFILES; exit(ACCESSCHANGE); end;
      ENDYET;if goodbye then exit(ACCESSCHANGE);
      seek(Questions,recno);
      get (Questions);                          (* accesschange cont'd*)
      with Questions^ do begin
        i:=0;
        if (recno =0) then begin
          writeln('  Number of questions is: ',count);
          write  ('  Enter correct number of questions: ');
          CHECKINPUT (count); questot:=count end;
```


BEST COPY

```
      repeat;
      CLRSCRN;
    GOTOXY(0,10);
      i:=i+1;
      phrase:=(material[i]);
      check:=copy(phrase,1,1);
      writeln(material[i]);
      writeln(dash,dash);
      writeln('  Type replacement for this line       (or <CR> if O.K.)');
      readln (material[i]);
      temporary:=(material[i]);
      place:=copy(temporary,1,1);
      if (length(temporary)=0) then material[i]:=phrase;
    until (LOOKEND(check)=1)or(LOOKEND(place)=1);
                                              (*accesschange cont'd*)
      if (LOOKEND(check)=1) and (length(temporary)>1) then begin
        repeat;
          i:=i+1;
          writeln('  Enter new line of text:       or <*>');
          readln(material[i]);
          phrase:=(material[i]);code:=copy(phrase,1,1);
          until (LOOKEND(code)<2);end;
        end;
    CLRSCRN;
    ENDYET;if goodbye then exit(ACCESSCHANGE);
    seek (Questions,recno);
    put (Questions);
    until (recno>=questot);
    CLOSEFILES;
  end;                          (*.............end of accesschange *)

procedure ACCESSREAD;
  var resp: string[1];                    (* primary routine to read file *)
  begin; CLRSCRN; DIRECTORY; SETFILENAME;writeln(filename);
    (*$I-*) reset (Questions, filename);
            reset (OUT,trk);
            iocheck:= ioresult;
    (*$I+*)
    if iocheck<>0 then begin writeln ('File not found  ...');
            SETFILENAME;
            reset (Questions,filename);
            reset (OUT,trk); end;
    recno:=1;
    readln (OUT,jtxt);
    while (recno>0) do begin;
        SEERECNUM;
      if (goodbye) then begin;CLOSEFILES;exit(ACCESSREAD);end;
        writeln(dash);
    ENDYET;if goodbye then exit(ACCESSREAD);
        seek (Questions, quesloc);          (* accessread cont'd*)
        get (Questions);
        with Questions^ do begin
              CLRSCRN;
              GOTOXY(0,1);
              i:=1;code:=' ';
```


BEST COPY

```
    while (not goodbye)and (LOOKEND(code)<>1) do begin;
          phrase:=(material[i]);
          code :=copy(phrase,1,1);
          writeln (material[i]);
          i:=i+1; end;
      ENDYET;if goodbye then exit(ACCESSREAD);
          seek (Questions, recno );
         get (Questions);
           i:=1;code:=' ';
    while (not goodbye)and(LOOKEND(code)<>1) do begin;
          phrase:=(material[i]);
          code:=copy(phrase,1,1);
          writeln (material[i]);
          i:=i+1; end;                                    (* accessread cont'd *)
 write ('.....   Press <RETURN> key when you have finished reading...');
     readln (resp);
     if (length(resp)>0)or goodbye then begin;
        CLOSEFILES;exit(ACCESSREAD);end;
    end;
  end;
end;                                  (*.....end of accessread *)

procedure  SHOWMENU;
  begin writeln (dash,dash,dash);              (* menu to screen *)
        writeln ('Enter the letter for the procedure you wish to use:');
        writeln;
        writeln ('  W -- write new test items ');
        writeln ('  C -- revise existing test items ');
        writeln ('  R -- review contents of a test ');
        writeln ('  Q      --  quit');
        writeln;
        writeln (dash,dash,dash);
        write ('  Your choice is ?   ');
 end;

procedure PRINTMENU;
  var checkcode : char; checker,entry:CS;
  begin; SHOWMENU;
     readln (checkcode);
     checker:=[checkcode];
     entry:=['W','C','R','Q','w','c','r','q'];
     if checker<=entry then keycode:= checkcode
     else begin writeln ('....Sorry, incorrect code...  Enter again: ');
                readln (keycode)
                end;
            writeln(dash,dash,dash,dash);
  end;

  BEGIN  (*.........................MAIN.................*)
  repeat;PRINTMENU;
    writeln(dash,dash,dash,dash);
      case keycode of
        'W','w':  ACCESSMAKE;
        'C','c':  ACCESSCHANGE;
        'R','r':  ACCESSREAD;
        end;
        until (keycode='q') or (keycode='Q');
        writeln ('.....................thanks  !');
        writeln (dash,dash,dash,dash);
   END.
```


BEST COPY

```
{$S+*) Program RUNTEST;                              (*   edition 15-apr-85  *)
rogram to administer one microcomputer-managed diagnostic testing session
     ...with upward and downward movement through test file ......
veloped at UCLA under the auspices of The National Institute of Education,
       Grant No. G-84-Ø112, P1 by D.L..McArthur and C.R.Morris.......*)
 CONST Maxinfo = 4;Maxquest = 25;Maxlines = 18;Maxitem = 25;
       Maxhead  = 9;Screensize= 22;qq='  -=-=-=-=-=-=-=-=-=-  ';
 TYPE item = record material    :array[1..Maxlines] of string[8Ø] end;
 VAR testresp :array [1..Maxquest] of integer;
     testinfo :array[1..Maxinfo,1..Maxquest] of integer;
     testhypo :array[1..Maxinfo,1..Maxquest] of integer;
   H0,H1,H2,H3,H4, NumofItem,Attempt,Correct,rightknt,wrongknt  :integer;
   howmany,recno, itemn, questn, leveln, quesloc, badresp  :integer;
   questnum,i,j,k,l,m,count,oldlevel,oldrecno,oldquesloc,oldcount  :integer;
   entrylevel,pause,furthest  :integer;
   BUFFER,newline,filename,testfile,phrase,trk  :string[8Ø];
   TH1,TH2,TH3,TH4,TH5, coder, Response, Answer  :string[1];
   ok  :string[6];
   keycode, code  :char;
   feedback,good,de,goup  :Boolean;
   Questions  :file of item;OUT :text;

ocedure NAME;
  var q:Boolean;begin;good:=false;
  write (' Please type your first and last name');
                      writeln('    and finish with the RETURN key');
     gotoxy(15,6);readln (phrase);if (length(phrase)>Ø) then begin;
     for i:=1 to length(phrase) do begin;
           q:=false;if phrase[i] in ['a'..'z'] then q:=true;
                if phrase[i] in ['A'..'Z'] then q:=true;
                if (phrase[i] in ['-',''''',' ']) then q:=true;
                if (phrase[i]=' ') then q:=true;
                if (q=false) then exit(NAME);end;
     good:=true;end; end;

ocedure FIRSTTEXT;
 begin; if (m<>Ø) then exit(FIRSTTEXT);. (* frontend text *)
 repeat;readln(OUT,phrase);writeln(phrase);
 until eof(OUT);for i:=1 to pause do begin;end;
 close(OUT,lock);end;

nction LOOKEND (code:string):integer;
 var foundit:boolean; index:integer;
 begin
   index:=1;foundit:=FALSE;
   while (not foundit) and (index <= Maxlines) do begin
     if (code = '*') then begin
       foundit:=TRUE;LOOKEND:=1 end
     else begin
       index :=index+1; LOOKEND:=Ø end
     end;
   if (code='?') then LOOKEND:=99;
nd;
```

BEST COPY

```
ocedure INITZERO;
 begin;m:=0;
 k:=sizeof(testinfo);fillchar(testinfo,k,0);
 k:=sizeof(testhypo);fillchar(testhypo,k,0);
 k:=sizeof(testresp);fillchar(testresp,k,0);
     questnum:=m;feedback:=true;de:=false;ok:='abcdef'; keycode:='a';
     Response:='q';
     recno:=m;oldlevel:=m;oldcount:=m;oldquesloc:=m;rightknt:=m;wrongknt:=m;
     furthest:=m;NumofItem:=m;
     group:=false;

     (*:::::::::user-defined parameters here:::::::::*)
     howmany:=4;
     entrylevel:=2;
     feedback:=true;end;


ocedure INITIALIZE;
 begin; WRITE(qq);GOTOXY(0,0);
   filename:='#4:STARTTEST.TEXT';(*$I-*)reset(OUT,filename)(*$I+*);
=ioresult;(*if (m=10)or(m=9)then begin;writeln;
             writeln(' WE NEED A FRESH DISK FOR DATA ...');
                                    exit(program);end; *)
INITZERO;
       gotoxy(0,2);WRITELN(' <><><><>  UCLA  <><><><>');
     NAME; if not good then begin;repeat;
              writeln ('Please try again:  FIRST  LAST  <return>');
              NAME; until good; end;
      coder:=copy(phrase,1,1);
      if (coder='Z')then de:=true; (*debugger*)
      if (coder='z') then de:=true;
           for j:=2 to 12 do begin(*2*);
      if (j<length(phrase))then coder:=copy(phrase,j,1);
          if (coder=' ') then i:=j+1;end(*2*);
         j:=length(phrase)-i+1;
             if (j>7) then j:=7;
             if j<1 then begin;
                 i:=1;j:=length(phrase);end;
                 coder:=copy(testfile,4,1);
              filename:=concat('#5:',copy(phrase,1,1),'.',
                   copy(phrase,i,j),'.',coder,'.TEXT');
   pause:=1;if (m<>10) then pause:=2000;writeln;
   FIRSTTEXT;end;

ocedure CLRSCRN;
 begin;GOTOXY(0,count);for i:=count to Screensize do begin;
       write('                                                  ');
       writeln('                                                  ');end;end;

ocedure COLLECTITEMDATA;
 var respnum: integer;
 begin ;respnum:=0;
   if Response[1]in ['a'..'e'] then respnum:=(pos(Response,ok));
    (*with iteminfo[recno] do begin; *)
      testinfo[1,recno]:=NumofItem;
      testinfo[2,recno]:=Attempt;
      testinfo[3,recno]:=Correct;
      testresp[recno]:=respnum;
      testinfo[4,recno]:=Leveln;
      testhypo[1,recno]:=H1;
      testhypo[2,recno]:=H2;
      testhypo[3,recno]:=H3;
      testhypo[4,recno]:=H4;
    end;(*end; *)
```

BEST COPY

```
ocedure FILOUT;
egin;phrase:=copy(filename,1,4);
          trk:=copy(filename,6,length(filename)-5);
                coder:='2';if (pos('2',filename)<>Ø) then coder:='3';
                if (pos('3',filename)<>Ø) then begin;
                     coder:='4';m:=1;end;
               filename:=concat(phrase,coder,trk);end;

ocedure SAVEDATA;(*with filechecking*)
 var  outfile:interactive;
   begin;    writeln(qq);
       if (furthest<1)then begin;close(outfile,purge);exit(SAVEDATA);end;
   count:=Ø;CLRSCRN;gotoxy(Ø,6);writeln(qq,'... Please wait ...',qq);
   repeat;
        (*$I-*) reset   (outfile,filename);m:=ioresult;(*I+*)
        if ( m =  Ø)then FILOUT;(* if file already exists*)
   until (m<>Ø);
        rewrite (outfile,filename); writeln(qq);
          writeln(outfile,'Testnum?=Ø, ');
          writeln(outfile,'Questnum?=Ø, ');
          writeln(outfile,'Attempt[(1) yes (2) no]?=Ø, ');
          writeln(outfile,'Correct[(1) yes (2) no]?=Ø, ');
          writeln(outfile,'Response[(1) a (2) b (3) c (4) d]?=Ø, ');
          writeln(outfile,'Level?=Ø, ');
          writeln(outfile,'HYP1[(1) yes (2) no]?=Ø, ');
          writeln(outfile,'HYP2[(1) yes (2) no]?=Ø, ');
          writeln(outfile,'HYP3[(3) yes (2) no]?=Ø, ');
          writeln(outfile,'HYP4[(1) yes (2) no]?=Ø* ');
       for i:=1 to furthest do
         (*with iteminfo[i] do *)begin;
           write (outfile, i:5);
           write (outfile, testinfo[1,i]:5);
           write (outfile, testinfo[2,i]:5);
           write (outfile, testinfo[3,i]:5);
           write (outfile, testresp[i]:5);
           write (outfile, testinfo[4,i]:5);
           write (outfile, testhypo[1,i]:5);
           write (outfile, testhypo[2,i]:5);
           write (outfile, testhypo[3,i]:5);
           writeln (outfile,testhypo[4,i]:5);
           end;
     writeln (outfile, '    Ø    Ø    Ø    Ø    Ø    Ø    Ø    Ø    Ø* ');
  close(outfile,lock);
  writeln(qq,filename,qq);CLRSCRN; end;

ocedure SETRECNUM;
 begin;read (OUT, recno, itemn, questn, leveln, quesloc,HØ,H1,H2,H3,H4);
f de then writeln(recno,' ', itemn,' ', questn,' ', leveln,' ', quesloc,' ',HØ,
 ' ',H1,' ',H2,' ',H3,' ',H4);
 if (goup=false) then BEGIN;IF DE THEN WRITELN( '$1');exit(SETRECNUM);END;
 if (goup=true) then begin;
     close(OUT,lock);
     reset(OUT,trk);
     readln(OUT,phrase);
     wrongknt:=Ø;rightknt:=Ø;
   repeat;if (recno<1)then exit(SETRECNUM);if eof(OUT)then exit(SETRECNUM);
          read(OUT, recno, itemn, questn, leveln, quesloc,HØ,H1,H2,H3,H4);
        if de then writeln(recno,' ', itemn,' ', questn,' ',
         leveln,' ', quesloc,' ',HØ,' ',H1,' ',H2,' ',H3,' ',H4);
       until (leveln<>level+1);end;
 end;
```


BEST COPY

```
ocedure WHATRESP;
 begin;
      good:=false;
      if Response= 'A' then Response :='a';
      if Response= 'B' then Response :='b';
      if Response= 'C' then Response :='c';
      if Response= 'D' then Response :='d';
      if Response= 'E' then Response :='e';
      k:=pos(' ',newline)-1;if k>0 then newline:=copy(newline,1,k);
      for k:=1 to (length(newline)-1) do begin;
         coder:=copy(ok,k,1);if (Response=coder) then good:=true;end;
              write(' ');end;

ocedure PROCESSRESP;
 begin;NumofItem :=NumofItem+1;
         if Response=TH1 then H1:=0;
                                  (*... here are the error hypos...*)
         if Response=TH2 then H2:=0;(* 2,3,4 not used *)
         if Response=TH3 then H3:=0;
         if Response=TH4 then H4:=0;
         if (length(Response)=0) then begin
           Attempt:=2;Correct:=0;end;
         if (length(Response)>0) then
            if (Response=Answer) then begin;
               Correct:=1;Attempt:=1;rightknt:=rightknt+1; end
            else begin;
               Correct:=2;Attempt:=1;wrongknt:=wrongknt+1; end;
               if de then writeln ('Process:::',Response,Attempt,Correct);
 if feedback then begin;gotoxy(0,count);
if (Response=Answer) then writeln(qq,'Answer ',Answer,' is CORRECT ',qq);
   if (Response<>Answer) then writeln (qq,'Sorry, the answer is ',Answer,qq);
   for k:=1 to pause do begin;end;(*pausing*)end;
 end;

ocedure TYPEYOU;
    begin;gotoxy(1,count);
rite('... Choose one letter for the best answer:    <',phrase,'>   ');
sponse:='g';unitclear(1);read(keycode);
sponse[1]:=chr(ord(keycode));Response:=copy(Response,1,1);
          WRITELN;end;

ocedure NEWLINER;
          begin;
          j:=length(newline)-1;phrase:='';
          for k:=2 to j do begin;
            if newline[k] in ['0','1','2','3','4','5','6']
                then phrase:=copy(ok,1,k);
                IF DE THEN WRITELN('$$$ ',NEWLINE,K,PHRASE);end;end;
```


BEST COPY

```
ocedure ACCESS;
 begin(*1*); if de then writeln('ACCESS BEGUN');
    trk:=concat(testfile,'.Q.TEXT');
    reset (Questions, testfile);
    reset (OUT,trk);
              readln(OUT,phrase);    (* ...reading header for command file..*)
              if de then writeln(' phrase : ',phrase);
    repeat;
          if eof(OUT) then exit(ACCESS);
          if de then writeln('setting recnum');
  SETRECNUM;good:=false;(* "gomore" used as flag for read in next repeat*)
 until leveln=entrylevel;
    IF DE THEN WRITELN(' COMMENCING REPEAT');
    repeat
         if de then writeln('beginning ifthens');
        if (recno>oldrecno)then furthest:=recno;
        oldrecno:=recno;
        if (oldlevel<>leveln) then wrongknt:=0;
        if (oldlevel<>leveln) then rightknt:=0;
        oldlevel:=leveln;
        if good   then SETRECNUM;
 (rightknt>=howmany) then if (leveln>1) then goup:=true;
    if (recno<1) then exit(ACCESS);(* ... no more questions remain ...*)
    if (wrongknt>=howmany) then if (leveln<3) then begin(*2*);
                while (leveln=oldlevel) do SETRECNUM;(*godown*);
                if (leveln<1) then exit (ACCESS);
                end(*2*);
        seek (Questions, quesloc);
        get (Questions);
    with Questions^ do begin(*4*);
      if (quesloc<>oldquesloc) then begin;    (*   new item stem?  *)
                count :=0; CLRSCRN;coder:=' ';
                GOTOXY(0,1);i:=1;
   while (LOOKEND(coder)<>1) do begin(*5*);
                  phrase:=(material[i]);
                  coder :=copy(phrase,1,1);
                  if (coder<>'*') and (coder<>')') then writeln (material[i]);
                  count:=i;
                  i:=i+1;
                  end(*5*);
             oldquesloc:=quesloc;oldcount:=count;end(*4*);
      if (quesloc=oldquesloc) then begin(*6*);
               count:=oldcount;
               CLRSCRN;end(*6*);
        i:=1;
        seek(Questions,recno );
        get (Questions);
          coder:=' ';badresp:=0;
          GOTOXY(0,count );
```

BEST COPY

```
          while (LOOKEND(coder)<>1) do begin(*7*);
            phrase := (material[i]);
            coder  := copy(phrase,1,1);
               if (LOOKEND(coder)=999) then begin(*8*);
                    count:=1;CLRSCRN;exit(ACCESS);end(*8*);
            if (coder <> '*') then writeln (material[i]);
             I:=i+1;end(*7*);
            count:=count+i-1; if (count>22) then count:=23;
          i:=i-1;newline := (material[i]); NEWLINER;
            1:=0;TYPEYOU;
          TH1:=copy(newline,2,1);
          TH2:=copy(newline,3,1);
          TH3:=copy(newline,4,1);
          TH4:=copy(newline,5,1);
          TH5:=copy(newline,6,1);
            if (TH1='0')then Answer:='a';
            if (TH2='0')then Answer:='b';
            if (TH3='0')then Answer:='c';
            if (TH4='0')then Answer:='d';
            if (TH5='0')then Answer:='e';
                WHATRESP;if not good then begin;
                  repeat; GOTOXY(0,count);
                  write ('  Try again .....');
                    badresp:=badresp+1;
                    1:=12;TYPEYOU;WHATRESP;
                    if (badresp>1)then good:=true;  until good ;end;
                PROCESSRESP;
          COLLECTITEMDATA;
 end(*2*);
       until (recno>=Maxquest ); end(*1*);

GIN (*.....................................main program.....*)
testfile:='#5:P';          (* pronoun test *)
INITIALIZE;ACCESS;
   close(Questions,lock);close(OUT,lock);SAVEDATA;

testfile:='#5:C';          (* comprehension test *)
          k:=pos('.P.',filename);phrase:=copy(filename,1,k);
          filename:=concat(phrase,'C.TEXT');
INITZERO;ACCESS;
   close(Questions,lock);close(OUT,lock);SAVEDATA;
            gotoxy(0,12);
            writeln(qq,qq,' Thank you ',qq,qq);
            gotoxy(30,22);writeln(filename);
END.
```


BEST COPY

 ©1985 Regents of the University of California

```
PROGRAM SUMMARY;        (* edition  20-mar-85  with Dx implemented *)
   var  tempted,  right, errorknt, j,k,toplev,botlev,iocheck    :INTEGER;
        testnum,questnum,attempt,correct,response,level,hyp  :integer;
        tempbylev,righbylev   :array [1..5] of integer;
        hypos       :array [1..10] of integer;
        filename, subj, qq,testtype   :string[40];
        phrase   :string[80];
        op, de,goodresp  :Boolean;
        INS,FO,ERRS :text;
        keycode  :char;

procedure ShallWeGo;
   begin;qq:=('-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-');writeln;writeln(qq);
           if (length(subj)<1) then exit(ShallWeGo);
j:=length(subj);for k:=1 to j do if subj[k] in ['a'..'z'] then
       subj[k]:=chr(ord(subj[k])-ord('a')+ord('A'));
           if subj='BYE' then exit(program);
           writeln(qq);writeln;
if pos(' DISK',subj)<1  then begin;
   k:=length(subj);for j:=2 to k do if subj[j]=' ' then subj[j]:='.';
            if pos('.TEXT',subj)=0 then subj:=concat(subj,'.TEXT');
     if pos('.C.',subj)>0 then testtype:=' comprehension test';
     if pos('.P.',subj)>0 then testtype:=' pronoun test';
                        writeln(qq,testtype,qq);
       filename:=concat('#5:',subj);
       if pos('.TEXT',filename)=0 then filename:=concat(filename,'.TEXT');
   if pos('..',filename)>0 then for k:=1 to length(filename)-1 do
        if filename[k]='.' then if filename[k+1]='.' then filename[k]:=' ';
   writeln('....looking for..... ',filename);writeln;
   (*$I-*)  reset (INS, filename);  iocheck:=ioresult;
      if iocheck<>0 then writeln('.....Sorry, that file isn''t found...');
      if iocheck=0 then goodresp:=true;  (*$I+*)
   end;
  end;

procedure Directry;
    var volume :integer;
       count  ,i,prog,byte,block,l,entry  :integer;
       buffer  :packed array [0..511] of 0..255;
       ptitle :string;
     begin; entry:=0;volume:=5;writeln;block :=2;byte:=6;iocheck:=-1234;
     unitread (volume,buffer,512,block,0);
     writeln;count:=buffer[16];
     writeln ('  Catalog of the test results on file');writeln;
     for prog:=1 to count do begin
       byte :=byte + 26;
       if byte >511 then begin;
         block :=block+1;byte:=byte-512;
         unitread (volume,buffer,512,block,0);end;
         ptitle := '                                ';
         L:=buffer[byte];
         if byte+L>511 then L:=511-byte;
         for i:=1 to L do
           ptitle[i] :=(chr(buffer[byte+i]));
           if pos('.TEXT',ptitle)>0 then begin;
             if pos('.0.',ptitle)=0 then write (ptitle);
             entry :=entry+1;if entry mod 3 =0 then writeln;end;
       end;
    end;
```


BEST COPY

```
procedure Getfile;
   begin;goodresp:=false;
 repeat; Directry;
   writeln;
   writeln (' Please type the name of the file to be summarized:');
   writeln ('            NAME  <return>');
   writeln ('                or insert a different data disk in the 2nd drive');
   writeln ('                                then hit <return>');
   writeln ('                or  type  BYE <return>');readln (subj);
ShallWeGo;
until goodresp;
  write ('            Do you want to print this student''s summary?  <Y/N>  ');
   op:=false;readln (phrase);
   if phrase[1]='Y' then op:=true;
   if phrase[1]='y' then op:=true;
   k:=length(subj)-7;phrase:=copy(subj,1,k);
          for j:=1 to 25 do writeln;gotoxy(0,0);
 writeln('     Computing summary for ',phrase,'  ',testtype);
   writeln('       ',qq);writeln;
   if op then begin;rewrite (FO,'printer:');
        writeln('  Please be sure your printer is ON');
        writeln(FO,'   Performance summary for ',phrase,': ',testtype);
        writeln(FO,'                           ',qq);writeln(FO);end;
 end;

procedure NAME;
   begin;
   tempted :=0; right :=0;toplev:=1;botlev:=5;
          for j:=1 to 5 do begin;
               hypos[j]:=0;tempbylev[j] := 0;righbylev[j] := 0;end;
   for k:=1 to 10 do begin;readln(INS,phrase);end;
   repeat;
      readln (INS, testnum,questnum,attempt,correct,response,level,hyp,k);
      if (k>hyp) then hyp:=k;
      if de then writeln( testnum,questnum,attempt,correct,response,level);
             (*Testnum?=0,
             Questnum?=0,
             Attempt[(1) yes (2) no]?=0,
             Correct[(1) yes (2) no]?=0,
             Response[(1) a (2) b (3) c (4) d]?=0,
             Level?=0,
             HYP1[(1) yes (2) no]?=0,
             HYP2[(1) yes (2) no]?=0,
             HYP3[(3) yes (2) no]?=0,
             HYP4[(1) yes (2) no]?=0*
                1    0    0    0    0    0    0    0    0*)
    if level>0 then if level <6 then begin;
      if level<botlev then botlev:=level;if de then writeln('$1');
      if level>toplev then toplev:=level;if de then writeln('$2');
      if attempt>0 then begin;
      tempted :=tempted+1 div attempt;if de then writeln('$3');
        tempbylev[level] := tempbylev[level] +1 div attempt;
        if de then writeln('$4');end;
      if correct>0 then begin;
      right := right+1 div correct;if de then writeln('$5');
        righbylev[level] := righbylev[level] +1 div correct;
        if de then writeln('$6');
        end;if de then writeln('$7');end;if de then writeln('$9');
               (* ... hypotheses .. *)
      if correct>1 then if hyp>0 then if hyp<6 then begin;
          hypos[hyp]:=hypos[hyp]+1;  (* tally error types *)
          hypos[hyp+5]:=testnum;     (*  and give item #  *)
             end;(* ... hypotheses .. *)
    until eof(INS);
  end;
```


BEST COPY

```
procedure TOPBOT;
begin;(*1*)
        for j:= toplev downto botlev do begin;(*2*)
 if tempbylev[j] >0 then righbylev[j]:=righbylev[j]*100 div tempbylev[j]
    else righbylev[j]:=-1 ;
   if de then writeln(j,'   ',tempbylev[j]);
   if righbylev[j] =-1 then begin;(*3*)
                writeln ( '      Level ',j,' not attempted');
         if op then writeln (FO,'      Level ',j,' not attempted');end;(*3*)
   if righbylev[j]<>-1 then if j=toplev then begin;(*4*)
 writeln (righbylev[j],' % correct at highest level attempted (',j,')');
 if op then writeln
 (FO,'      ',righbylev[j],' % correct at highest level attempted (',j,')');
                      end;(*4*)
   if righbylev[j]<>-1 then if j=botlev then begin;(*5*)
 writeln (righbylev[j],' % correct at lowest level attempted (',j,')');
 if op then begin;
writeln(FO,'      ',righbylev[j],' % correct at lowest level attempted (',j,')')
;
                      writeln(FO);end;end;(*5*)
   if righbylev[j]<>-1 then if j>botlev then if j<toplev then begin;(*6*)
        writeln (righbylev[j],' % correct at level ',j);
        if op then writeln (FO,'      ',righbylev[j],' % correct at level ',j);
                      end;(*6*)
     end;(*2*)
  end;(*1*)

procedure Performance;
 begin;gotoxy(2,1);writeln('                                ',qq,qq,qq);writeln;
  writeln('    Out of ',tempted,' items attempted  , ',right,
                    ' items were correct');
if op then writeln(FO,'    Out of ',tempted,' items attempted  , ',right,
                    ' items were correct');
        if tempted>0 then right := right*100 div tempted;
        writeln ('                    ',right,' % correct overall');
      if op then writeln (FO,'                    ',right,' % correct overall');
        writeln;writeln;
        TOPBOT;close(INS,lock);end;

procedure Diagnose;
 begin;
  writeln;if right>99 then exit(Diagnose);writeln;
  writeln(' This student made ----');
  if op then writeln(FO,' This student made ----');
  filename:='#4:ERRORS';
     if pos('.C.',subj)>0 then filename:=concat(filename,'.C');
     if pos('.P.',subj)>0 then filename:=concat(filename,'.P');
  filename:=concat(filename,'.O.TEXT');
  reset(ERRS,filename);
  for j:=1 to 5 do begin;
  right:=hypos[j];
  k:=hypos[j+5];
    readln (ERRS,phrase);
    if hypos[j]=1 then begin;
       writeln ('A single error in ',phrase);
       writeln ('                  It occurred in Item # ',k);
                if op then
       writeln (FO,'A single error in ',phrase);
   if op then writeln (FO,'                  It occurred in Item # ',k);
            END;
    if hypos[j]>1 then begin;
       writeln (right,' errors in ',phrase);
   writeln ('                  An example of this is Item # ',k);
```

33 BEST COPY

```
                if op then
    writeln (FO,right,' errors in ',phrase);
if op then writeln (FO,'                    An example of this is Item # ',k);
                END;
  end;writeln(qq);if op then writeln(qq);close(ERRS,lock);
 end;

procedure Continue;
  begin;gotoxy(1,23);
write('.....Hit <return> to continue,                or type BYE <return> ');
        readln(subj);ShallWeGo;end;

BEGIN;
  de:=false;
repeat;Getfile; Name; Performance; Diagnose;Continue;
if op then begin;writeln(FO);writeln(FO);writeln(FO);close(FO,lock);end;
           until subj='BYE';END.
```

Sample text showing test item layout

Reading Comprehension Test

There are many active volcanoes on earth such as the famous Mauna Loa in Hawaii. They occasionally shoot smoke and lava onto the earth's surface. The formation of a volcano can be thought of as an underground balloon buried under thin layers of sand and plaster. Hot magma fills the balloon, and pushes up the ground. Magma is the hot material that forms the earth's center core. The magma pushes upward inside the volcano and melts the surrounding rock and dirt. The melted rock and dirt is called lava. Eventually, the lava will push through the top of the mountain. This action causes a volcanic explosion and forms a crater. After the explosion, or eruption, the lava cools off and the mountain sides shrink somewhat. This is similar to a balloon shrinking after it pops. A volcano is said to be inactive when it no longer produces eruptions.

* ← Flag:end of Item text

This passage is mostly about how...

a. volcanos are filled with lava.
b. mountains are sometimes volcanoes.
c. volcanoes are created.
d. magma creates mountains.

*3303 ← Flag:end of Question text plus response markers (i.e. Error types: "a" = 3, "b" = 3, "c" = correct response, "d" = 3

An active volcano...

a. erupts every other month.
b. erupts occasionally.
c. never erupts.
d. contains cold lava.

*2022

According to the passage, when a balloon pops, its sides...

a. shrink.
b. become larger.
c. crack.
d. become rounder.

*0111

) ← Flag: end of last item

Butterflies are created through an interesting series of stages. Adult butterflies lay their eggs on leaves or tree branches. These eggs turn into caterpillars. The caterpillars crawl around the trees and bushes, feeding on leaves. Eventually, the caterpillars weave themselves a silken shell called a cocoon. The caterpillars sleep inside the cocoon for a few weeks. Inside the cocoon, the caterpillar grows its butterfly wings, antennae and body. When it awakens, the butterfly breaks out of its cocoon and flies away. When it lays eggs, the process will begin again.

*

The passage is mostly about how...

a. adult butterflies are cute.

BEST COPY

Sample location cue files

Question number	Difficulty level	Location for Question	Check value	Location for Item stem	Error type Answer "a"	Answer "b"	Answer "c"	Answer "d"	Answer "e"	
12	This is the command sequence for #5:P									pronoun test 4/17/85
	1	1	1	3	26	3	3	3	3	0
	2	1	2	3	26	2	2	2	0	2
	3	1	3	3	26	1	1	1	1	0
	4	1	4	3	26	3	3	3	3	0
	5	1	5	3	26	1	1	0	1	1
	6	1	6	3	26	0	2	2	2	2
	7	2	8	2	27	2	2	2	0	2
	8	2	9	2	27	0	3	3	3	3
	9	2	10	2	27	1	1	0	1	1
	10	2	11	2	27	1	1	0	1	1
	11	2	12	2	27	2	2	0	2	2
	12	2	13	2	27	3	3	0	3	3
	13	3	15	1	28	0	2	2	2	2
	14	3	16	1	28	0	3	3	3	3
	15	3	17	1	28	1	1	1	1	0
	16	3	18	1	28	3	0	3	3	3
	17	3	19	1	28	1	0	1	1	1
	18	3	20	1	28	2	2	2	2	0
	0	0	0	0	0	9	9	9	9	9

Question number	Difficulty level	Location for Question	Check value	Location for Item stem	Error type Answer "a"	Answer "b"	Answer "c"	Answer "d"	Answer "e"	
12	This is the command sequence for #5:C									comprehension test 4/17/85
	1	1	1	3	26	3	3	0	3	9
	2	1	2	3	26	2	0	2	2	9
	3	1	3	3	26	0	1	1	1	9
	4	2	5	3	27	0	3	3	3	9
	5	2	6	3	27	2	2	0	2	9
	6	2	7	3	27	1	1	0	1	9
	7	3	9	2	28	0	2	2	2	9
	8	3	10	2	28	1	0	1	1	9
	9	3	11	2	28	3	0	3	3	9
	10	4	13	2	29	0	3	3	3	9
	11	4	14	2	29	2	2	2	0	9
	12	4	15	2	29	1	1	0	1	9
	13	5	17	1	30	3	0	3	3	9
	14	5	18	1	30	2	2	2	0	9
	15	5	19	1	30	0	1	1	1	9
	16	6	21	1	31	2	0	2	2	9
	17	6	22	1	31	1	0	1	1	9
	18	6	23	1	31	3	0	3	3	9
	0	0	0	0	0	9	9	9	9	9

BEST COPY

Sample test performance summaries

```
   Performance summary for D.MACA:  pronoun test
                          -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    Out of 11 items attempted  , 3 items were correct
                   27 % correct overall
      33 % correct at highest level attempted (2)
      20 % correct at lowest level attempted (1)

 This student made ----
3 errors in using nominative pronoun.
                   An example of this is Item # 17
2 errors in using direct object pronoun.
                   An example of this is Item # 11
3 errors in using pronoun as object of preposition.
                   An example of this is Item # 16



   Performance summary for D.MACA:  comprehension test
                          -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    Out of 6 items attempted  , 5 items were correct
                   83 % correct overall
      0 % correct at highest level attempted (3)
      100 % correct at lowest level attempted (2)

 This student made ----
A single error in literal comprehension of a sentence.
                   It occurred in Item # 1



   Performance summary for M.BACCUS:  pronoun test
                          -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    Out of 12 items attempted  , 9 items were correct
                   75 % correct overall
      66 % correct at highest level attempted (2)
      83 % correct at lowest level attempted (1)

 This student made ----
A single error in using nominative pronoun.
                   It occurred in Item # 10
A single error in using direct object pronoun.
                   It occurred in Item # 7
A single error in using pronoun as object of preposition.
                   It occurred in Item # 16



   Performance summary for M.BACCUS:  comprehension test
                          -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

    Out of 7 items attempted  , 5 items were correct
                   71 % correct overall
      100 % correct at highest level attempted (3)
      66 % correct at lowest level attempted (2)

 This student made ----
A single error in identifying the main idea
```


BEST COPY